Supporting Application Network Flows with Multiple QoS Constraints

Amit Mondal
Northwestern University
Evanston, USA
a-mondal@northwestern.edu

Puneet Sharma, Sujata Banerjee
Hewlett-Packard Labs
Palo Alto, USA
{puneet.sharma, sujata.banerjee}@hp.com

Aleksandar Kuzmanovic
Northwestern University
Evanston, USA
akuzma@cs.northwestern.edu

***Abstract*—There is a growing need to support real-time applications over the Internet. Real-time interactive applications often have multiple quality-of-service (QoS) requirements which are application specific. Provisioning QoS in the Internet through IP routing — Intserv or Diffserv — faces many technical challenges, and is also deterred by the huge deployment issues. As an alternative, application providers often build their own application-specific overlay networks to meet their QoS requirements.**

In this paper, we present a unified framework which can serve diverse applications with *multiple* QoS constraints. Our scalable flow route management architecture, called *MCQoS*, employs a hybrid approach using a path vector protocol to disseminate aggregated path information combined with on-demand path discovery to find paths that match the diverse QoS requirements. It uses a distributed algorithm to dynamically adapt to an alternate path when the current path fails to satisfy the required QoS constraints. We do large-scale simulation and theoretical analysis to show that our approach is both efficient and scalable, and that it substantially outperforms the state of the art protocols in accuracy. Our simulation results show that MCQoS can reduce the false negative percentage to less than 1% compared with 5-10% in other approaches, and eliminates false positives, whereas other schemes have false positive rates of 10-20% with minimal increase in protocol overhead. Finally, we implemented and deployed our system on the Planetlab testbed for evaluation in a real network environment.

I. Introduction

There is a growing need for supporting real-time applications over the Internet. Real-time and interactive applications often have multiple quality-of-service (QoS) requirements which are application-specific. For example, multimedia streaming has high bandwidth requirements, whereas voice-over-IP does not need as much bandwidth but requires low delay. However, the best-effort service model of packet delivery in today's Internet is destination based, and does not distinguish between different flow requirements. Packets for different applications may have varied QoS requirements, but traverse the same Internet path to a destination. This often results in degraded performance for real-time and interactive multimedia applications, and gives poor network utilization. Although there have been advances in the QoS provisioning in the Internet through Intserv and Diffserv, these are far from being realized, as these models require significant changes in the Internet infrastructure. One-hop source routing has also been proposed for improving reliability of Internet paths [1]. Moreover, end-point based application specific overlay networks have been designed for specific applications, including multicasting [2], and content distribution networks [3]. However, one-hop source routing or end-user overlays typically cannot meet end-to-end QoS guarantees, since flows normally traverse many uncontrolled intermediate domains. As the multimedia and voice applications over the Internet are becoming increasingly popular, the service providers are deploying their own application-specific overlay networks.

Application-specific overlays have been further evolved into Service overlay networks (SONs) [4], [5] where each overlay node not only provides application-level data forwarding but also a set of value-added services (*e.g.* media transcoding). SONs are being designed to serve various Internet applications with a diverse set of QoS requirements. In a SON, overlay nodes are deployed all over the Internet across multiple ASes at strategic locations. User flows are relayed through a number of these SON nodes to meet their specific QoS constraints. Sometimes, SON providers sign up Service Level Agreements (SLA) with the underlying IP level service providers to obtain certain service guarantees.

One of the key challenges in SONs is to scalably find paths that satisfy the QoS constraints that the application requests. Previous work [6], [7] on QoS routing in overlay networks either considers a single QoS metric or combines multiple QoS constraints into a single composite metric, and uses a modified version of Dijkstra's algorithm to find a QoS path. This causes unpredictable and poor performance of QoS-sensitive applications like VoIP or video conferencing calls. Moreover, the problem of finding a route that satisfies multiple QoS constraints is NP-hard [8]. Current approaches to the multi-constrained path problem ([9]–[20]) rely on approximation and randomized algorithms.

In this paper, we present a unified framework which can serve diverse applications with multiple QoS constraints. It always admits the right number of flows, meets individual QoS constraints, and thus ensures their performance. Our contribution is three-fold. First, we design a scalable QoS routing protocol which employs a hybrid approach of path vector routing and on-demand path discovery. It minimizes the overhead of update messages in overlay networks by propagating only the *best* path for a destination for *each* QoS metric. It also utilizes an on-demand route discovery mechanism when flows cannot be admitted or rejected based on local path table information. Second, we propose a distributed algorithm to dynamically divert to an alternate path when the

current path violates QoS constraints. Finally, we do large-scale simulation and analysis to show that our approach is accurate, efficient, and scalable when compared with existing approaches. We have also implemented and deployed our system on the Planetlab testbed (http://www.planet-lab.org) for further evaluation in a real-world dynamic network setup. Our approach allows for higher utilization of network resources by decreasing the false negative ratio from 5-10% in other approaches, to less than 1% with minimal increase in protocol overhead. It also completely eliminates false positives while other schemes have false positive rates of 10-20%.

The rest of the paper is organized as follows. In Section II, we discuss the related work on this topic. Section III gives an overview of the overlay flow QoS management architecture followed by the design goals and challenges. In Section IV, we present our multi-constraint QoS route computation protocol. We analyze and evaluate the performance of the protocol in Section V. In Section VI, we present a prototype implementation of the protocol, followed by our conclusions in Section VII.

II. Related Work

In the last few years, many researchers have looked into the problem of reliability and QoS provisioning in the Internet through overlay networks. Resilient Overlay Network (RON) is an end-user based overlay network where nodes actively examine the conditions of the Internet paths between themselves and recover from path failures [21]. Application-specific overlay networks have been designed for many applications including multicasting [2] and content distribution networks [3]. The goals are to efficiently distribute content to a large number of users. However, end-user overlays cannot meet end-to-end QoS guarantees, as flows normally traverse many uncontrolled intermediate domains.

To address the end-to-end QoS problem, the concept of service overlay network is advocated [4]. Overlay nodes are deployed at strategic locations spreading over multiple ASes to form a service overlay backbone. Li and Mohapatra designed a QoS-aware routing protocol for overlay networks (QRON) [6]. QRON adopts a hierarchical methodology to enhance its scalability. However, QRON depends upon Dijkstra's algorithm for QoS path computation and works with a single composite metric. This makes QRON inaccurate and hence unsuitable when a flow has multiple QoS constraints and each metric has to be met individually.

Lao *et al.* proposed a distributed QoS routing protocol for backbone overlay networks referred to as QSON [7]. In QSON, probes loaded with QoS constraints are forwarded on all possible paths to the destination. Each probe computes the QoS parameter of the path it traverses. Finally, when the probes reach the destination the QoS parameters of all possible paths are computed and a QoS path is selected based on the information. However, the ad-hoc approach of QSON routing protocol incurs high overhead and increases the flow admission time significantly.

Lui *et al.* [22] propose a novel approach for topology aggregation in delay-bandwidth sensitive networks. It uses line segments in the delay-bandwidth plane instead of points to represent the QoS parameters of logical links. Tam *et al.* [23] use line segment representation in the distance vector routing protocol to address the scalability issue in multi-constraint QoS routing. In this approach, instead of receiving alternative path information, a node receives a line segment approximating the region of feasible QoS requests from each neighbor. It then aggregates the line segments received from all the neighbors, and forwards the aggregated line to other neighbors. However, this approach suffers from information loss and distortion which may lead to either over-estimation or under-estimation of path metrics. We compare the performance of our approach with the line-segment-based approach for aggregration of path information.

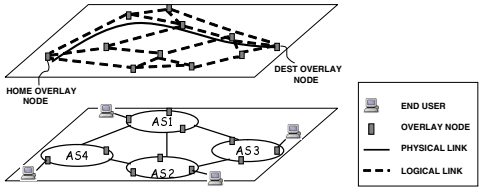

Fig. 1. Service overlay backbone model

Gu *et al.* [5] proposes a QoS-assured Service Composition (QSC) algorithm based on Dijkstra's algorithm to find the best path to serve users' requests. The authors propose a single metric to represent "cost" of each edge in the graph. For each user request, the algorithm generates a candidate graph that is comprised of all overlay nodes that provide any component-service from the service template. The path selection problem then involves finding the shortest path in the candidate graph. However, because this approach uses a composite metric, the path might not meet the individual QoS constraints.

Motiwala *et al.* proposed the path splicing approach [24] to enable end systems to construct paths by combining multiple routing trees to each destination over a single network topology based on their requirements. OverQoS [25] is another architecture for providing Internet QoS using overlay networks. The *controlled loss virtual link* (CLVL) abstraction in OverQoS ensures statistical rate and loss assurances to a single flow as long as the aggregate rate and loss meet the specification. Clearly, OverQoS cannot provide end-to-end QoS to flows under multiple metrics. To the best of our knowledge, there is currently no scalable dynamic overlay QoS routing protocol that finds a path under multiple constraints which individually meets each constraint.

III. Background: Flow QoS Management Challenges

We first give a brief overview of the overall flow QoS management architecture. Figure 1 illustrates the system. The underlying physical network shows a generic network topology comprised of four autonomous systems (ASes) and physical links among them. End users are shown connected to the network. The figure shows possible locations of overlay nodes. Each AS has multiple overlay agents. More overlay agents can be added incrementally to exploit all potential paths between any source and destination pair.

The higher overlay layer shows a network comprised of overlay nodes connected via logical links. Each node has sensors that measure logical link metrics to its direct overlay neighbors. This information is disseminated amongst the overlay nodes efficiently and scalably.

Each end user node is connected to a home overlay node. To set up an overlay path to a destination node, an end user sends a QoS flow setup request to its home overlay node. The home overlay node finds a feasible path from the home agent to the destination overlay node. A feasible path is a path which satisfies all QoS metrics for the flow. Once a feasible path is found, all intermediate overlay nodes are configured for per-flow forwarding of the data packets. End-to-end session performance is monitored, and path adaptation is triggered if the current path fails to meet the QoS metrics of the flow.

A. Design Challenges

We discuss here the challenges in designing the flow route management architecture to support applications with multiple QoS requirements in overlay networks.

Multiple QoS constraints. QoS path computation under multiple QoS constraints is a hard problem. The complexity of the path computation algorithm increases as the number of nodes increase. Finding a feasible path under multiple QoS constraints using Dijkstra's algorithm has been proven to be a *NP-hard* problem [8]. One alternative approach to reduce the complexity is to use heuristics and approximation algorithms. Another alternative approach is to work with a single composite metric which is derived from multiple metrics. However, the paths thus obtained might not meet each QoS requirement individually. Hence, we avoid relying upon Dijkstra's algorithm to find a QoS path in our routing protocol.

Dynamic overlay-link properties. In QoS routing, nodes need to know the paths' QoS properties which may vary dynamically. Therefore, as path properties change new update messages need to be sent. This increases the routing message overhead. Moreover, in an overlay network, overlay link properties change very frequently due to uncontrolled flow arrival and departure in the underlying IP network, which significantly increases the control message overhead. In order to scale with the network size the protocol needs to minimize the control message overhead. But at the same time, to use the network resources efficiently, the protocol should be able to discover a feasible QoS path to the destination in question, if one exists. Our protocol achieves both the conflicting objectives by adopting a hybrid approach of proactive routing and reactive routing.

Adaptive routing. Another challenge in overlay QoS routing is to continually monitor individual flows' performance to detect violation of QoS constraints and, in such cases, to adopt an alternate path to guarantee QoS. To scale with the number of flows in the network, our protocol uses a distributed algorithm to find and dynamically start using a new alternate path when needed.

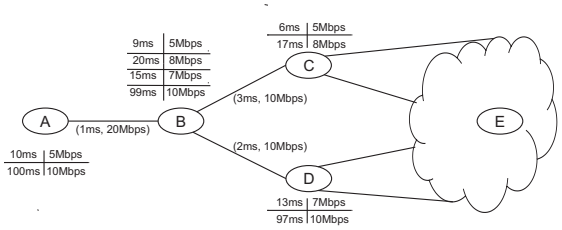

Fig. 2. Illustration of the MCQoS protocol through an example

IV. Overview of the Multi-Constraint QoS Protocol

In this section we discuss the step-by-step design of the Multi-Constraint QoS (MCQoS) routing protocol. We describe the various components of the MCQoS protocol in the context of an example scenario.

A. Path information dissemination

The MCQoS routing protocol is built upon the path vector routing protocol. In path vector routing, each node advertises reachability information to its neighbors, and forwards only the *best* path vector information for each reachable destination. This greatly reduces the overhead of path information dissemination and makes it possible to scale with network size.

However, path aggregation in a path vector protocol is a non-trivial problem when multiple metrics are associated with each path. The challenge is how to choose the *best* path to a destination from multiple alternative paths. The *best* delay path might not be the *best* bandwidth path and vice versa. Advertising all alternative paths for each destination is certainly not a scalable approach. Also it is possible that a QoS request cannot be served by either the best delay or best bandwidth path, but there exists a path which meets both delay and bandwidth requirements individually. For example, assume the path characteristics of the best delay path from source to destination are (10ms, 5Mbps), and the best bandwidth path has path characteristics (100ms, 10Mbps). A request with QoS requirement of (20ms, 8Mbps) is not satisfied by either the best delay or the best bandwidth path.

Thus, we adopt a hybrid approach of path vector routing protocol combined with on-demand route discovery. We modify the path vector protocol such that nodes advertise the best path for *each* QoS metric instead of a single best path for each destination[1]. Moreover, the path vector advertisements are now tagged with the QoS metrics of the path.

Upon reception of a QoS request a node searches its local path table for a feasible path, and admits or rejects the flow depending upon the existence of a feasible path. However, there are cases when it is not possible to make a decision based on its local path table. It then invokes the on-demand route discovery module of the MCQoS protocol, which we explain in detail through an example in Section IV-B.

[1]Instead of sending only the best path, a design alternative would be to forward $k (k > 1)$ best paths for each QoS metric. This would potentially reduce the time taken for on-demand route discovery but at the cost of increased overhead for path information dissemination.

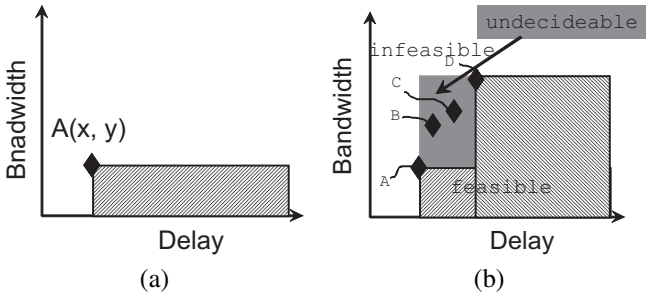

Fig. 3. Path aggregation with multiple QoS metrics

Consider the simple overlay topology shown in Figure 2. For simplicity of discussion, we consider only two QoS metrics – delay and bandwidth – though our approach is generalizable for any number of metrics. The tuple associated with the edge represents the QoS metrics of each logical link. Also, portions of local QoS tables, which include information applicable to the destination node E, are shown for each overlay node. For example, the table associated with node C indicates that C knows two paths to E with QoS (6ms, 5Mbps) and (17ms, 8Mbps).

Each overlay node aggregates and advertises the best paths to its neighbors for each reachable destination. Thus, node C learns about the best paths to the destination E, and path metrics for delay and bandwidth as: (6ms, 5Mbps) and (17ms, 8Mbps). Node C then advertises the best delay path and the best bandwidth path to its neighbor node B. Similarly, B learns about the best delay path and the best bandwidth path through node D. B then combines the received path advertisements with the link metrics to the corresponding neighbors, and computes all alternative paths with their corresponding delay and bandwidth.

In the figure, the QoS table at B shows that B knows about four alternative paths to E with QoS parameters $(9ms, 5Mbps)$, $(20ms, 8Mbps)$, $(15ms, 7Mbps)$ and $(99ms, 10Mbps)$ through its neighbors C and D. However, B advertises only the best delay path and the best bandwidth path to A. Thus, A learns about two alternate paths to E through neighbor B with QoS $(10ms, 5Mbps)$ and $(100ms, 10Mbps)$.

B. Route discovery in absence of a feasible path in local path table

As shown in Figure 2, node A does not know about all alternative path information because B advertises only the best path for each metrics. Thus, if a feasible path cannot be identified in its local path table then a node may request more information from its neighbors for alternate paths that were not disseminated because they were not the best paths for individual metrics. This information may then be used to find a feasible path. Next, we explain how a node decides when to send a request for extra path information.

Figure 3(a) shows QoS metrics for delay and bandwidth on a delay-bandwidth plane. Point $A(x\,ms, y\,Mbps)$ represents a path to the destination. A feasible request is a request that can be supported by a path. Feasible requests that can be supported by path A are requests in the area with $delay \geq x\,ms$ and $bandwidth \leq y\,Mbps$. Feasible requests that can be supported by the set of paths are the union of the areas supported by the individual paths.

Figure 3(b) shows all four paths from A to E on a delay-bandwidth plane. Points $A(10\,ms, 5\,Mbps)$, $B(16\,ms, 7\,Mbps)$, $C(21\,ms, 8\,Mbps)$ and $D(100\,ms, 10\,Mbps)$ each represents a path to the destination to E from A.

It is now obvious that since B advertises only the best delay and the best bandwidth path, A knows about only paths A and D (in Figure 3(b)). Thus, there are feasible requests which could be accepted but A cannot *admit* them based on its local path table. We call this region the *undecideable region* (shaded area in Figure 3(b)). However, node A can admit requests in the hatched region based on its local path table which we term as the *feasible region*. Moreover, A can also *reject* requests in the *infeasible region*, based on its own path table.

Algorithm IV.1: FINDQOSPATH(t, D^{QoS}, W^{QoS})

for each *neighbor* **n** *who advertised path to destination* **t**

do
- **comment:** Check if the request is in feasible region
- **if** $((((BestDelayPath(n,t).D + delay(s,n)) < D^{QoS})$ **and** $(min(BestDelayPath(n,t).W, bw(s,n)) > W^{QoS}))$ **or** $(((BestBWPath(n,t).D + delay(s,n)) < D^{QoS})$ **and** $(min(BestBWPath(n,t).W, bw(s,n)) > W^{QoS})))$
  - **then**
    - **comment:** There exists a feasible path
    - *return success*;

for each *neighbor* **n** *who advertised path to destination* **t**

do
- **comment:** Check if the request falls in infeasible region
- **if** $(((BestDelayPath(n,t).D + delay(s,n)) > D^{QoS})$ **or** $(min(BestBWPath(n,t).W, bw(s,n)) < W^{QoS}))$
  - **then**
    - **comment:** No feasible path
    - *continue*;
- **comment:** Check if the request is in undecidable region
- **if** $(((BestDelayPath(n,t).D + delay(s,n)) < D^{QoS})$ **and** $(min(BestBWPath(n,t).W, bw(s,n)) > W^{QoS}))$
  - **then**
    - **comment:** Undecidable region
    - $SendQuery(n, t, (D^{QoS} - delay(s,n)), W^{QoS})$

C. On-demand route discovery

The MCQoS protocol uses on-demand route discovery to decide whether to admit or reject a request when it falls in the undecideable region. Note that for requests in the feasible and infeasible region, a decision can be taken based on the local path table. A node queries its neighbors about the existence of a path for QoS requests in the undecideable region. Unlike on-demand routing protocols, such as Ad-Hoc On-demand Distance Vector Routing [26], where a node broadcasts path discovery requests to all neighboring nodes to find a valid path to a destination, in the MCQoS protocol, dissemination of route discovery messages are guided by the node's local QoS tables. It queries only those neighbors who advertised reachability information about the destination in question. If the node received reachability information from multiple neighbors, then instead of sending path discovery requests to all of them, it sends the path discovery request to only

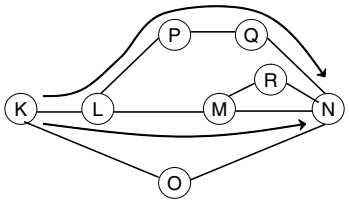

Fig. 4. Dynamic path adaptation in MCQoS

those neighbors, about which the node cannot make a decision about the (non)existence of a feasible QoS path through them. Algorithm *IV.1* depicts the pseudo code of the complete QoS path finding algorithm. It finds a QoS path from local node **s** to destination node **t** with delay constraint D^{QoS} and bandwidth constraint W^{QoS}.

D. Dynamic path adaptation

Dynamic adaptive routing is an important component of our overlay flow route management protocol. As there is no resource reservation, and new flows are admitted in the underlying IP network, overlay link properties may change uncontrollably. This means that the selected path of an admitted flow might fail to meet the QoS requirements over time. We propose a distributed algorithm for dynamic path adaptation.

In our protocol, each intermediate overlay node in the path knows the QoS requirements from the node to the destination. This information is propagated by flow setup messages, which are sent to all intermediate overlay nodes for flow-based routing. Also, an upstream node periodically pushes the QoS requirements to its downstream node.

For example, as the delay of the link carrying packets to next relay node changes, the upstream node forwards the delay requirement to its downstream node, and so on. Hence, each intermediate overlay node knows the most recent QoS constraints of the path from itself to the destination. This simplifies the path restoration problem and allows distributed path adaptation. As a node detects that the downstream path is failing to meet the delay, bandwidth, or loss constraints, it triggers an alternate path search to the destination at the local node. If the node finds such an alternate path to the destination in its path table that meets the constraints, then it switches to that path and informs all upstream and downstream nodes in the earlier path. Otherwise, it triggers an alternate path search at the immediate upstream node. This continues until the request reaches the source node or an alternate path is found.

Here we explain the distributed dynamic path adaptation protocol through an example. Consider the topology of overlay nodes K-R shown in Figure 4. The overlay nodes learn about the paths to other nodes and the associated metrics using path vector protocol. When K receives a path request to N with bandwidth requirement W^{QoS} and delay bound D^{QoS}, K searches its own QoS table to find a path which meets the constraints. Assume that K finds a path through L and M which meets both bandwidth and delay constraints. K then forwards a flow setup request with bandwidth requirement W^{QoS} and delay bound $(D^{QoS} - delay(KL))$ to L, and so on.

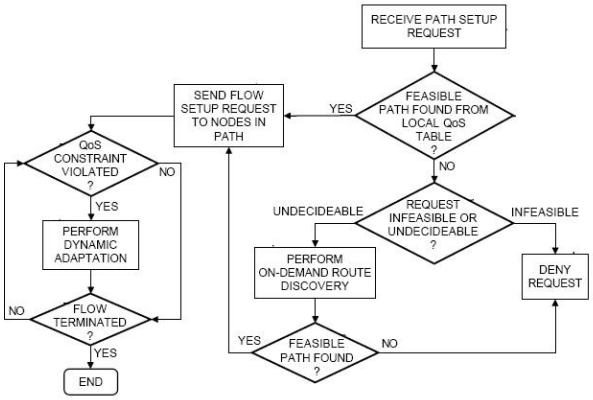

Fig. 5. Protocol flow chart

Once the flow has been admitted with path K-L-M-N, suppose M notices that the overlay link M to N no longer meets the bandwidth and/or delay constraints. M then searches its local QoS table for an alternative path to N which meets the constraints. Assume that the only alternative path M-R-N does not satisfy the constraints. Then M notifies its immediate upstream overlay node in the path, which is L. This triggers path discovery at L, and assuming that L already knows an alternative path through P that satisfies all the constraints, L sends a flow setup request to P. This enables quick dynamic path adaptation in a distributed fashion. If L cannot find a path that meets the constraints, then the process is repeated with its immediate upstream node, and so on until a decision is made.

One important aspect of any dynamic QoS routing protocol is the route instability problem. Changing network conditions may lead to path oscillations resulting in network instability. However, techniques like route dampening [27] can help to alleviate the route instability problem.

E. Putting it all together

Figure 5 illustrates the complete functioning of the protocol. A path setup request is received at an overlay node indicating QoS constraints for a flow. The path setup request identifies the source and destination for the path.

The overlay node searches its QoS path table to determine whether a path that satisfies the constraints exists. A path satisfying the constraints is a feasible path. If a path can be identified from the local table that is feasible, a flow setup request is sent to all the overlay nodes in the path, and the path for the flow is created.

If a feasible path is not found, the node determines whether the path request is infeasible or if the path request is undecideable. For example, as described in Figure 3, the node determines whether the request constraints fall within the infeasible region. If the node determines that the path is infeasible, a deny path message is sent to the node trying to create the path.

If the node determines that the path request is undecideable (i.e., falls within the undecideable region), on-demand route discovery is performed. This is where the node sends route discovery messages to its neighbors to identify other paths

that may not have been advertised. A determination is made on whether a feasible path can be found through the on-demand route discovery. If yes, a flow setup request is sent to all the overlay nodes in the path, and the path for the flow is created. If not, then a deny request message is sent to the node trying to setup the flow path and the request is rejected.

After the path for the flow is created, all the intermediate overlay nodes in the path, which are any overlay nodes between the source and destination, monitor path metrics for links to determine whether the QoS constraints for the flow are violated. If any QoS constraints are violated, then dynamic adaptation is performed to identify an alternative path, as described with respect to Figure 4. In some scenarios, there may be no new paths available that satisfy the constraints. Then, a message is sent indicating the current path is failing to meet the QoS constraints and no new path can be identified that satisfies the QoS constraints. Monitoring of link metrics and dynamic adaptation are performed until the flow is terminated.

V. Analysis and Evaluation

In this section, we analyze and evaluate the MCQoS protocol with respect to its scalability, overhead, accuracy and convergence time.

A. Protocol overhead analysis

Overheads associated with the route information dissemination and discovery are key to the scalability of the overlay QoS design. As discussed earlier, the MCQoS protocol adopts a hybrid approach of proactively disseminating path vectors for each QoS metric and then performing on-demand discovery for undecidable requests. We first discuss the overhead of the proactive part of the MCQoS protocol. Consider the simple overlay topology of six nodes as shown in Figure 6. For ease of explanation we consider only one QoS metric. The associated link weight represents the available bandwidth of the corresponding overlay link. In the path vector protocol, each node aggregates the received path information from multiple neighbors with local link QoS, and disseminates the best path to all its neighbors except the node from which it received the path information. Figure 7 shows all possible dissemination paths for the destination advertised by "1". The number of paths increases exponentially with the number of nodes in the topology and the out-degree distribution of the nodes. However, since the QoS path vector protocol advertises only the best path to a destination, some of the branches are pruned. For the example graph, the branches in the grayed region are pruned. The routing message overhead and convergence time can be computed from the tree. The convergence time is proportional to the length of the longest branch in the tree. The routing message overhead is proportional to the number of edges in the tree.

Here, we discuss the overhead of the reactive part of the MCQoS protocol. Two important factors that affect the overhead of the on-demand route discovery protocol are: (i) out-degree distribution of the nodes i.e. the average number of neighbors and (ii) overlay distance distribution from source to destination in the overlay graph. In the worst possible case, the overhead of route discovery is proportional to the sum of all

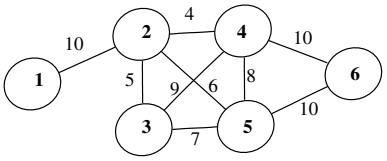

Fig. 6. Sample graph

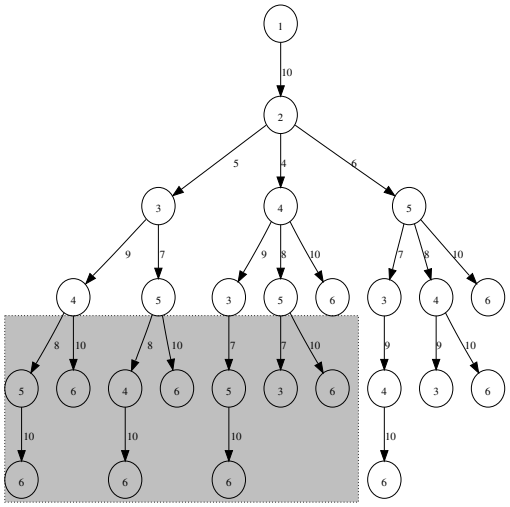

Fig. 7. Dissemination tree for the sample graph

possible path lengths from source to destination. However, the amortized cost of route discovery depends on what fraction of QoS falls in the undecidable region and the average number of hops it travels before a decision is made. Let h represent the average number of hops a route discovery message travels, and let β represent the average number of neighbors of the nodes. Then, the bandwidth overhead of a single route discovery request is proportional to $\beta \times h$.

This worst case scenario for route discovery happens when the route discovery request travels up to the immediate upstream node of the destination along the longest cycle-free path. In such a scenario the QoS request falls in the undecidable region for all the intermediate nodes. Let $d(u, v)$ represent the longest loop free path from source u to destination v in the graph. Then the route discovery message travels $d(u, v) - 1$ overlay hops along the longest path before it sends either a success or a failure message. It must be noted that the overhead of the on-demand route discovery can be further controlled by limiting the number of hops a discovery query is forwarded. In our simulations, more than 95% of the undecideable region could be discovered within five hops.

B. Experimental evaluation

We implement an event-driven simulator to capture the dynamics of the protocol in large-scale environments. We use GT-ITM [28] to generate a random flat topology of overlay nodes. The average outdegree of the nodes is $min(10, size/2)$. Then, we randomly map these nodes onto actual nodes on the Planetlab testbed and assign link metrics (delay, bandwidth) values from corresponding link measurement data in S^3 [29]. This enables us to evaluate the protocol performance in a realistic overlay network scenario.

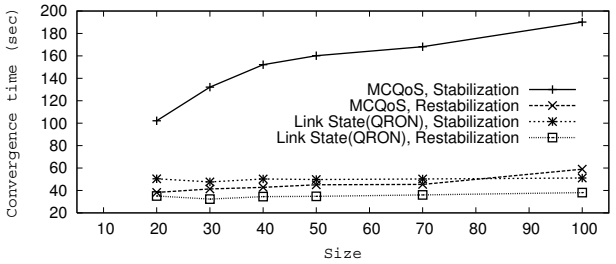

Fig. 8. Convergence time with advertisement interval = 10s

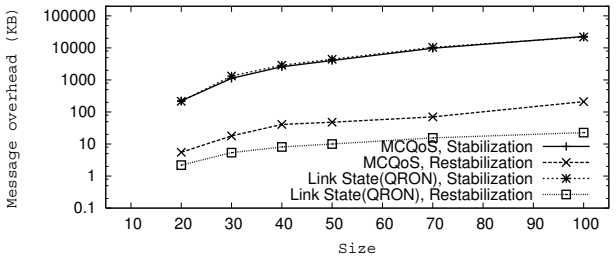

Fig. 9. Routing message overhead of the MCQoS protocol

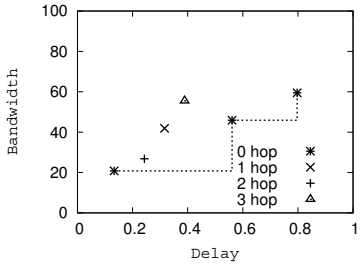

Fig. 10. Depletion analysis

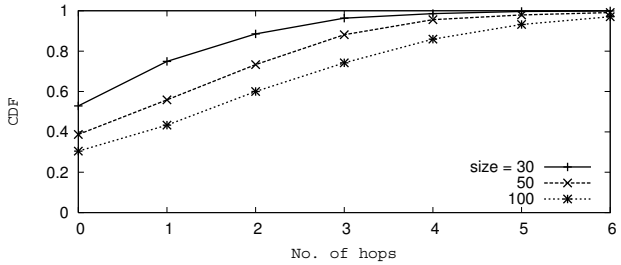

Fig. 11. CDF of number hops to discover the entire depletion area

Convergence time. How long does it take for a network of any given size to stabilize? We simulate the protocol behavior over a network snapshot, and measure the initial convergence time. The convergence time is expected to increase with the increase in number of overlay nodes in the overlay network. However, the increase should be sublinear to scale with network size. Also once the network has stabilized, how long does it take to restabilize if the link metrics of a certain link change? To measure this, we first let the network stabilize, change link metrics for a randomly chosen link, and then measure the reconvergence time.

Figure 8 shows the stabilization and restabilization time as a function of network size. The nodes advertise the aggregated path information periodically, and the advertisement interval is set to $10\,sec$ in all the experiments. The figure shows that initially the convergence time increases quickly as the network size increase but the growth plateaus once size reaches 50.

On the other hand, the re-convergence time remains almost constant (Figure 8) as network size increases. This is simply because as the number of nodes in the graph increases, the number of alternative paths between any two nodes also increases. Thus the probability that a certain link comprises the best path also decreases, hence the average reconvergence value decreases with increase in network size.

Figure 8 also compares the convergence times for MCQoS with link-state based approaches such as QRON [6]. Being a path-vector based protcol, MCQoS takes longer to converge and restablize, but it does not involve any NP-Hard computations and can scale better with network size.

Message overhead. Routing message overhead is another important metric for an overlay network to scale with network size. Figure 9 shows the total routing update message overhead as a function of network size. The plot shows that the message overhead increases with increase in network size. This is primarily because an increase in the number of nodes in the network means more and bigger size routing messages among nodes. But at the same time, the resources in the network increase with the number of nodes. Due to similar reasons, the message overhead for reconvergence also increases.

Figure 9 also shows the message overhead for QRON like protocols. The message overhead incurred by MCQoS is comparable to the link-state based approaches.

Depletion area. As the path vector protocol forwards only the best path set, inlcuding only the best path for each metric, to its neighbors, many alternative paths are suppressed along the way. Thus, the feasible QoS region known at a node for a destination may be smaller than the "global feasible region". We use the term global feasible region for the feasible QoS region at the source node if the source node knew all alternative paths to the destination like link-state protocols. We explain this via Figure 10. We term paths in the undecidable region of the source node which are discovered within k hops, of the on-demand discovery process, as "k hop" paths. Thus, only "0 hop" paths are known at the source node. The figure plots the new paths that are found as discovery proceeds away from the source. The figure shows that for the given example, there is only one new "1 hop" path that is in the undecidable region of the source node and discovered within one hop distance. There is also one "2 hop" path which is in the undecidable region of the source node and two hops away from the source node. The new bandwidth-delay region covered by the new path indicates how much depleted region is dicovered at each hop.

We use the term *depletion area* to represent the global feasible QoS region which is not known to the source node. We measure the number of hops it takes to discover the entire depletion area for a given source destination pair. Figure 11 plots cumulative distribution of the number of hops traversed to discover the entire depletion area for different network sizes. The figure shows that almost the entire depletion area is discovered within 6 hops from the source node. Nearly 35% of the times, the source node knows the global feasible region for a network size of 50 nodes. However, this decreases as the topology increases in size. This is an important evaluation criteria as it determines how to set the parameter for restricting on-demand discovery without impacting the accuracy.

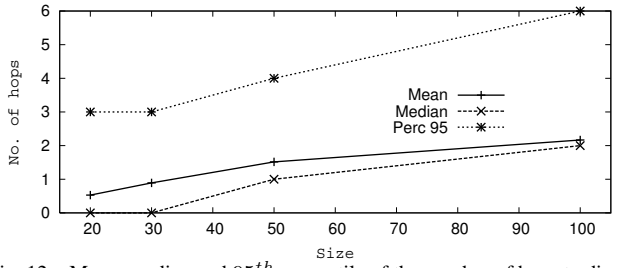

Fig. 12. Mean, median and 95^{th} percentile of the number of hops to discover the entire depletion area

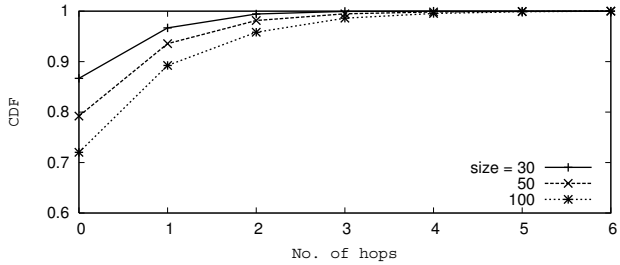

Fig. 13. CDF of depletion area discovered as a function of number of hops

Figure 12 plots the mean, median, and 95^{th} percentile of the number of hops to discover the entire depletion area as a function of network size. The figure shows that the average number of hops needed increases linearly with the increase in network size. Moreover, the median is always less than the mean, which indicates that in most cases it takes less than the average number of hops to discover the entire feasible region. Finally, the 95^{th} percentile line indicates that we can even restrict the route discovery process within five hops, and discover more than 95% of the depletion area.

We also measure the fraction of the depletion area discovered within k hops from the source node. Figure 13 plots the cumulative distribution of the depletion area discovered at each hop for different network sizes. The plot indicates that more than 90% of the depletion area is discovered within one hop from the source node.

Note that the message overhead incurred by the on-demand discovery of undecideable paths in MCQoS depends on the hops traversed by these requests. Hence the above plots also give an upper bound on the overhead of the on-demand route discovery protocol to find the QoS path between an arbitrary pair of source destination nodes.

Comparison with alternative approaches. The QoS requirement associated with each QoS request is distributed as follows: bandwidth between [5Mbps, 55Mbps] and delay between [100ms, 400ms]. A flow is admitted or rejected based upon composite metric cost of the end-to-end path, and the QoS request. We evaluate the accuracy of a simple composite metric approach like QSC [5]. The formula used to compute the composite metric value is similar to the formula used in Cisco's IGRP protocol [30] which is $(k_1 * delay + k_2/bw)$, where $delay$ is in seconds and bw is in bps. The values for the weights are: k_1 = 1 , and k_2 = 10^7.

Figure 14 shows the false positive (where a flow is admitted but the path does meet in individual QoS criteria), and false negative (a QoS path exists but the flow is not admitted) ratios as a function of network size. The false positive ratio

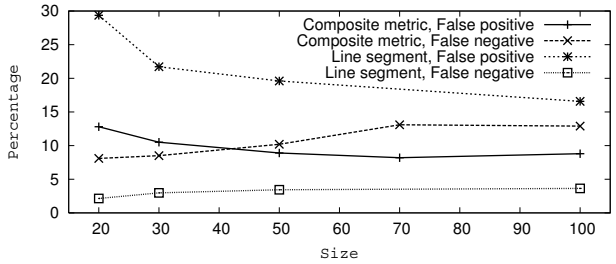

Fig. 14. False positive and False negative with alternative approaches

shows a decreasing trend, whereas false negative shows an increasing trend as network size increases. For a network size of 50 nodes, the false positive and false negative ratios are around 9% and 10%, respectively. We do believe that these percentage values can be improved by choosing application-specific composite metric. But our point here is to show that a feasible path with composite metric may not satisfy individual QoS metrics, and vice versa. On the other hand, our hybrid approach has no false positives and the false negative ratio can be easily reduced to less than 1% (Figure 13) by using only 3-hop on-demand route discovery.

Lui *et al.* [22] proposed a novel approach for topology aggregation which uses line segment(s) in the delay-bandwidth plane instead of points to represent the QoS parameters of a set of logical links. Later, Tam *et al.* [23] uses this approach to design a distance-vector based QoS routing protocol. However, this approach suffers from information loss and/or distortion, causing it to sometimes accept infeasible requests or reject feasible requests.

We measure the false positive and false negative ratio of their approach in terms of fraction of the undecidable region which is actually infeasible but their approach labels the area as feasible (false positive), and fraction of undecidable region which is actually feasible but is labeled as infeasible (false negative). Instead of implementing the entire protocol, we approximate the performance. We fit a regression line with each alternate path from the source to the destination as a point on the delay-bandwidth plane, and measure the false positive and false negative ratios as defined above. Figure 14 also shows the performance of the line-segment based approach. For a network of size 50 nodes, the false positive ratio is more than 20%, and false negative ratio is around 5%.

Arrival rate (conn/min)	60	120	240	300	600
Violation ratio (%)	0.32	0.33	0.78	0.4	1.12

TABLE I
QoS VIOLATION RATIO

Violation ratio in a dynamic environment. To evaluate MCQoS in dynamic environment where flows arrive and depart we do the following simulation. We generate a 100 node network, and generate QoS flow setup requests at a certain rate. Each flow lasts from 5 to 10 minutes. The QoS associated with each QoS requests are distributed as follows: bandwidth between [5Mbps, 55Mbps] and delay between [100ms, 400ms]. We simulate the network behavior for 10m̃in. As new QoS requests arrive before the network stabilizes after a flow is admitted we expect to observe QoS violations for subset of admitted flows. Table I shows the violation

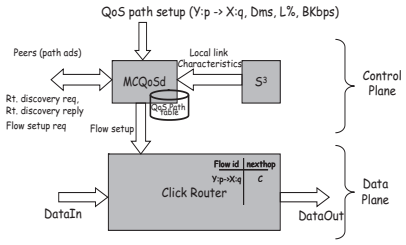

Fig. 15. System architecture

ratio, measured as the ratio of the flows that experience QoS violation over the total admitted flows, for different flow arrival rates.

VI. System Implementation

We built an initial prototype in C++ on a Unix platform. Figure 15 shows the overall system architecture. We use the Open Source implementation of BGP [31] and modified it to implement our protocol. The implementation uses S^3 [29] for sensing overlay link properties. The data forwarding plane was developed using the Click Modular Routing platform [32]. We enhanced Click to support per-flow forwarding. Our prototype, MCQoSd, receives QoS path setup requests from users in the form $(saddr,\ sport,\ daddr,\ dport,\ delay,\ loss,\ bw)$, uses MCQoS to find a path. If a feasible path exists, it configures all the intermediate overlay nodes along that path for per-flow forwarding.

We deployed the system in 50 Planetlab nodes, and set up a random flat overlay topology using GT-ITM [28]. We restricted the system to only one successful measurement of QoS metrics to its direct overlay peers, and let it stabilize. The deployment was able to discover paths for video streaming with QoS constraints similar to those used for simlation. Our future work involves systematic evaluation of its performance in dynamic network environments.

VII. Conclusion

Overlay QoS routing is an alternative to QoS IP routing for provisioning QoS in the current Internet. In this work, we designed a scalable QoS flow route management protocol for overlay networks using a hybrid approach of path vector routing and on-demand route discovery. The step-by-step path computation approach of path vector routing shields the protocol from complex, computationally intensive graph search algorithms to find a feasible path under multiple QoS constraints with link state routing. Moreover, the hybrid approach keeps a balance between flow setup time and control message overhead and greatly increases the scalability of the protocol. We performed large-scale simulations and theoretical analysis that demonstrated the efficiency and scalability of our approach.

References

[1] K. Gummadi, H. Madhyastha, S. Gribble, H. Levy, and D. Wetherall, "Improving the reliability of Internet paths with one-hop source routing," in *USENIX OSDI*, San Francisco, CA, Dec. 2004.

[2] Y. Chu, S. Rao, and H. Zhang, "A case for end system multicast," in *ACM SIGMETRICS*, Santa Clara, CA, Jun. 2000.

[3] B. Krishnamurthy, C. Wills, and Y. Zhang, "On the use and performance of content distribution networks," *ACM SIGCOMM Workshop on Internet Measurement*, pp. 169–182, Nov. 2001.

[4] Z. Duan, Z. Zhang, and Y. Hou, "Service overlay networks: SLAs, QoS, and bandwidth provisioning," *IEEE/ACM Transactions on Networking*, vol. 11, no. 6, pp. 870–883, 2003.

[5] X. Gu, K. Nahrstedt, R. Chang, and C. Ward, "QoS-assured service composition in managed service overlay networks," in *IEEE ICDCS*, Providence, RI, May 2003.

[6] Z. Li and P. Mohapatra, "QRON: QoS-aware routing in overlay networks," *IEEE Journal of Selected Areas in Communications*, vol. 22, no. 1, Jan. 2004.

[7] L. Lao, S. S. Gokhale, and J.-H. Cui, "Distributed QoS Routing for Backbone Overlay Networks," in *Networking*, Coimbra, Portugal, May 2006.

[8] Z. Wang and J. Crowcroft, "Quality-of-service routing for supporting multimedia applications," *IEEE Journal of Selected Areas in Communications*, vol. 14, no. 7, pp. 1228–1234, 1996.

[9] T. Korkmaz and M. Krunz, "A randomized algorithm for finding a path subject to multiple QoS requirements," *Computer Networks*, vol. 36, no. 2-3, pp. 251–268, 2001.

[10] X. Yuan, "Heuristic Algorithms for Multiconstrained Quality-of-Service Routing," *IEEE/ACM Transactions on Networking*, vol. 10, no. 2, 2002.

[11] G. Xue and W. Zhang, "Multiconstrained QoS Routing: Greedy is Good," in *IEEE GLOBECOM*, Washington, DC, Nov. 2007.

[12] L. K. A. McAuley, K. Manousakis, "Flexible QoS Route Selection with Diverse Objectives and Constraints," in *IEEE IWQoS*, Enschede, The Netherlands, Jun. 2008.

[13] G. Xue, A. Sen, W. Zhang, J. Tang, and K. Thulasiraman, "Finding a path subject to many additive QoS constraints," *IEEE/ACM Transactions on Networking*, vol. 15, no. 1, pp. 201–211, 2007.

[14] G. Liu and K. Ramakrishnan, "A* Prune: an algorithm for finding K shortest paths subject tomultiple constraints," in *IEEE INFOCOM*, Anchorage, Alaska, Apr. 2001.

[15] P. Van Mieghem and F. Kuipers, "Concepts of exact QoS routing algorithms," *IEEE/ACM Transactions on Networking*, vol. 12, no. 5, pp. 851–864, 2004.

[16] C. Pornavalai, G. Chakraborty, and N. Shiratori, "Routing with multiple QoS requirements for supporting multimedia applications," *Telecommunication Systems*, vol. 9, no. 3, pp. 357–373, 1998.

[17] G. Xue, W. Zhang, J. Tang, and K. Thulasiraman, "Polynomial time approximation algorithms for multi-constrained QoS routing," *IEEE/ACM Transactions on Networking*, vol. 16, no. 3, pp. 656–669, 2008.

[18] S. Chen, M. Song, and S. Sahni, "Two Techniques for Fast Computation of Constrained Shortest Paths," *IEEE/ACM Transactions on Networking*, vol. 16, no. 1, pp. 105–115, 2008.

[19] F. Kuipers, P. Van Mieghem, T. Korkmaz, and M. Krunz, "An overview of constraint-based path selection algorithms for QoS routing," *IEEE Communications Magazine*, vol. 40, no. 12, pp. 50–55, 2002.

[20] R. Hou, K. Lui, K. Leung, and F. Baker, "An Approximation Algorithm for QoS Routing with Two Additive Constraints," in *IEEE ICNP*, Orlando, FL, Oct. 2008.

[21] D. Anderson, H. Balakrishnan, M. Kaashoek, and R. Morris, "Resilient overlay networks," in *ACM SOSP*, Banff, Canada, Oct. 2001.

[22] K. Lui, K. Nahrstedt, and S. Chen, "Routing with topology aggregation in delay-bandwidth sensitive networks," *IEEE/ACM Transactions on Networking*, vol. 12, no. 1, pp. 17–29, 2004.

[23] W. Tam, K. Lui, S. Uludag, and K. Nahrstedt, "Quality-of-Service routing with path information aggregation," *Computer Networks*, vol. 51, no. 12, pp. 3574–3594, 2007.

[24] M. Motiwala, M. Elmore, N. Feamster, and S. Vempala, "Path Splicing," in *ACM SIGCOMM*, Seattle, WA, Aug. 2008.

[25] L. Subramanian, I. Stoica, H. Balakrishnan, and R. Katz, "OverQos: An overlay based architecture for enhancing internet Qos," in *USENIX NSDI*, Berkeley, CA, Mar. 2004.

[26] C. Perkins, E. Royer *et al.*, "Ad hoc On-Demand Distance Vector (AODV) Routing," in *IEEE Workshop on Mobile Computing Systems and Applications*, 1999.

[27] C. Villamizar, R. Chandra, and R. Govindan, "BGP Route Flap Damping," 1998, RFC 2439.

[28] K. Calvert and E. Zegura, "GT Internetwork Topology Models (GT-ITM)." [Online]. Available: http://www.cc.gatech.edu/projects/gtitm/

[29] P. Yalagandula, P. Sharma, S. Banerjee, S. Basu, and S. Lee, "S3: a scalable sensing service for monitoring large networked systems," in *ACM SIGCOMM workshop on Internet network management*, Pisa, Italy, Sep. 2006.

[30] "IGRP Metric." [Online]. Available: http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a008009405c.shtml/

[31] "The Openbgpd Project," http://openbgpd.org/.

[32] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. Kaashoek, "The click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, 2000.